TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de)
http://db.in.tum.de/teaching/ss21/ei2/

Blatt Nr. 5.5

1 Fakultät

Vervollständigen Sie die *rekursive* Methode, die für eine Zahl n die Fakultät berechnet.

```
int factorial(int n) { // für n > 0
    if (n==1) return 1;
    return n * factorial(n-1);
}
```

2 Fibonacci Zahlen

Vervollständigen Sie die *rekursive* Methode, die die n-te Fibonacci Zahl berechnet.

```
int fibonacci(int n) { // für n > 0
    if (n == 1 || n == 2) return 1;
    return fibonacci(n-1) + fibonacci(n-2);
}
```

3 Stufen

Angenommen, ein Kind kann eine Treppe mit s Stufen erklimmen, indem es entweder $1, 2, ..., k$ Stufen in einem einzigen Schritt nimmt. Berechnen Sie alle unterschiedlichen Schrittfolgen, mit denen das Kind genau die s Stufen der Treppe erklimmt. Vervollständigen Sie den Programmcode:

```
void stufen(int verbleibendeStufen, List<Integer> bisherigeSchritte, int k) {
    if (verbleibendeStufen == 0) { // moegliche Loesung gefunden
        for (Integer i : bisherigeSchritte)
            System.out.print(i + " - ");
        System.out.println("");
    }
    for (int step = 1; step <= k; step++) {
        if (step > verbleibendeStufen) break;
        bisherigeSchritte.add(step);
        stufen(verbleibendeStufen - step, bisherigeSchritte, k);
        bisherigeSchritte.remove(bisherigeSchritte.size() - 1);
    }
}
```

Tipp: In Java werden Listen genauso wie alle anderen Objekte mit Call by Reference aufgerufen – was bedeutet das für Ihre Methode?

4 Türme von Hanoi [anspruchsvoll]

In dieser Aufgabe implementieren wir das bekannte Knobelspiel „Die Türme von Hanoi“.[1].

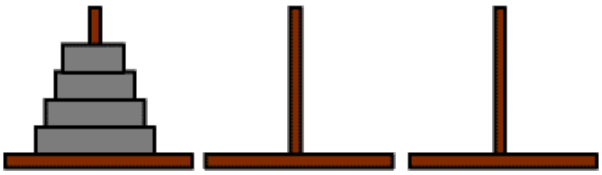

Abbildung 1: Die Türme von Hanoi mit vier Scheiben

Überlegen Sie sich einen rekursiven Algorithmus um alle Scheiben von Position 1 auf Position 3 zu verschieben. Dabei darf immer nur die oberste Scheibe eines Turms auf einen anderen Turm gelegt werden, wenn dessen oberste Scheibe größer ist (oder an der Zielposition gar keine Scheiben sind). **Tipp:** Damit die unterste Scheibe zur Position 3 bewegt werden kann, muss der darüber liegende Turm erstmal auf Position 2 verschoben werden. Dann kann man die größte Scheibe verschieben und dann den kleineren Turm von Position 2 auf die große Scheibe verschieben. Überlegen Sie sich wie dieses Prinzip auch für das Verschieben der kleineren Türme zum Tragen kommt.

```
// Diese Methode nimmt die oberste Platte von source und fügt sie oben zu dest
↪  hinzu
void moveDisk(Stack<Integer> source, Stack<Integer> dest) {
    assert(source.peek() < dest.peek());
    dest.push(source.pop());
}

void hanoi(int numPlates, Stack<Integer> source, Stack<Integer> buffer,
↪  Stack<Integer> destination) {
    if (numPlates == 0) return;
    hanoi(numPlates - 1, source, destination, buffer);
    moveDisk(source, destination);
    hanoi(numPlates - 1, buffer, source, destination);
}
```

[1] Für eine ausführliche Beschreibung siehe Wikipedia: `http://de.wikipedia.org/wiki/Türme_von_Hanoi`. Das ganze kann auch auf unzähligen Seiten ausprobiert werden, z.B.: `http://vornlocher.de/tower.html`